Exonet : sauvegarde automatisée d'une base de données

Description

Propriétés	Description
Intitulé long	Sauvegarde automatisée d'une base de données
Formation concernée	BTS Services informatiques aux organisations
Matière	SISR4 – Administration des systèmes
Présentation	Cet exercice invite à étudier un script PERL puis à l'améliorer pour conserver la trace du traitement des sauvegardes et faciliter la surveillance du système. L'exécution de ce script est programmée chaque soir.
Notions	Langage de commande, scripting.
Transversalité	Développement d'application et génie logiciel ; Architecture matérielle et logicielle.
Outils	Un ensemble de logiciels libres disponibles sous Windows, Linux ou MacOS qui peut être installé en une seule passe : Le PGI OpenERP, le SGBD PostgreSQL, un interpréteur Python. Un interpréteur PERL. Vous trouverez les documentations fonctionnelle, technique et utilisateur du PGI OpenERp en suivant ce lien : http://doc.openerp.com/
Mots-clés	PGI, ERP, OpenERP, PostgreSQL, script, PERL, sauvegarde.
Durée	4h.
Auteur(es)	Marie-pascale Delamare avec l'équipe SI du réseau CERTA
Version	v 1.0
Date de publication	Mai 2008

Le contexte de travail

La société Zenta vient de mettre en place le PGI OpenErp. Ce PGI s'appuie sur une base de données unique, hébergée par le système de gestion de base de données relationnelle Postgres.
Actuellement stagiaire dans l'entreprise Zenta, votre mission est de sauvegarder chaque soir et de façon automatique cette base de données.
Pour faciliter votre travail, l'administrateur réseau vous a fourni une ébauche de script Perl que vous trouverez en **Annexe 1.**

Travail à faire
1 – Commenter chaque ligne de l'ébauche de script Perl « sauvegarde.pl », fournie en annexe 1.

Gestion d'un fichier log

Vous voulez améliorer ce script pour imprimer dans un fichier log le résultat de chaque étape, de façon à faciliter la surveillance du système chaque matin. Le nom de ce fichier log : « log.txt », est passé en paramètre au script Perl.

Travail à faire
2 – Ajouter dans le script Perl la gestion de ce fichier de log (passage du nom par paramètre, ouverture, écriture des messages de réussite et fermeture). Vous utiliserez l'annexe 2.

Test des codes retour

Après un premier test qui a échoué par manque d'espace sur le disque pour créer le fichier dump, vous décidez de tester les codes de retour de chacune des étapes avant d'écrire un message de réussite ou d'échec dans le fichier log.

Pour vérifier l'étape de sauvegarde, vous devez tester le code retour du programme « pg_dump ». Si ce programme s'est bien déroulé vous écrivez le message « La sauvegarde s'est bien déroulée » dans le fichier « log.txt ». Si cette première étape s'est mal déroulée, vous inscrivez un message d'échec dans le fichier « log.txt » et vous interrompez le traitement.

Pour vérifier l'étape de transfert, vous testerez le code retour de la commande « ftp ». Si cette commande s'est bien déroulée vous écrivez le message « Le transfert s'est bien déroulé » dans le fichier « log.txt ». Si cette deuxième étape s'est mal déroulée, vous inscrivez un message d'échec dans le fichier « log.txt ».

Travail à faire
3 – Ajouter dans le script Perl la gestion des codes de retour et des messages de réussite ou d'échec de chacune des étapes du traitement. Vous utiliserez l'annexe 3.

L'analyse des erreurs

Pour faciliter l'analyse en cas d'erreur, vous souhaitez rediriger les résultats du programme « pg_dump » et de la commande « ftp » dans les fichiers « pg_dumplog.txt » et « ftplog.txt ». Ces fichiers sont détruits chaque fois qu'une étape s'est correctement déroulée. Pour détruire un fichier on utilise la commande « del ».

Travail à faire
4 – Ajouter dans le script Perl la gestion des fichiers de log de chacune des étapes.

Planifier le traitement

Vous souhaitez déclencher l'exécution de ce script chaque soir à 23 h 00.

Travail à faire
5 – Planifier ce traitement. Vous utiliserez l'annexe 4.

Annexe 1 : Ébauche de script Perl « sauvegarde.pl »

```
# ! /usr/bin/perl
use strict ;
#
my $date = (localtime)[7] ;
#
system "pg_dump -i -h localhost -p 5432 -U tinypg -F c -v -f \"sauvegarde.dump.$date\" zenta ;
#
system “ftp -i -n -v -s:ftp.txt”
#
```

La variable « $date »

La variable $date contient le numéro dans l’année du jour d’exécution.

Le programme « pg_dump.exe »

La sauvegarde d’une base de données Postgres s’effectue à l’aide du programme « pg_dump.exe ». Pour exécuter ce programme, il faut lancer la commande suivante :

```
pg_dump.exe -i -h localhost -p 5432 -U tinypg -F c -v -f "sauvegarde.dump" zenta
```

-f file : nom du fichier de sortie ;
-h host : serveur hébergeant la base postgres ;
-p port : port sur lequel écoute la base postgres ;
-U user : nom de l’utilisateur ;
-F format : format du fichier de sortie (c : fichier exploitable par le programme pg_restore et déjà compressé) ;
-v verbose : fichier log bavard ;
-i ignore : on ignore les divergences de versions ;
-zenta : nom de la base de données.

L’instruction « system »

On ordonne au système d’exploitation via l’instruction « system » de lancer une commande.

Le client « ftp »
–i : enlève le mode interactif,
–n : indique qu’il faut aller chercher le compte et le mot de passe sous lequel se connecter dans un fichier.
–v : donne un mode « bavard »
–s : indique dans quel fichier se trouvent les commandes ftp à exécuter.

Le contenu du fichier ftp.txt
open ftp.zenta.fr
user zenta
zenta
*put sauvegarde.dump.**
quit

Annexe 2 : Passage de paramètres et gestion de fichiers en Perl

Récupérer un argument passé en paramètre à un script Perl :

Exemple d'appel de script Perl
C:\>sauvegarde.pl log.txt

Récupération du paramètre passé dans le script Perl
my $log = $ARGV[0];

Gestion de fichiers en Perl :

Ouverture d'un fichier en écriture :
open (F, "> $log") || die " Problème à l'ouverture du fichier : $!" ;

Ecriture dans un fichier :
print F "La sauvegarde commence\n";

Fermeture de fichier :
close F

Annexe 3 : Test de code retour et éléments de syntaxe Perl

Test d'un code retour

Pour tester le code retour d'une commande en Perl, il suffit de tester la variable $?. Si cette dernière contient zéro alors la commande s'est bien déroulée.

L'écriture d'un « si » en Perl :

```
if ($?= =0)
        {
        Bloc d'instructions à exécuter si la condition est vraie ;
        }
        else
        {
        Bloc d'instructions à exécuter si la condition est fausse ;
        }
```

Annexe 4 : Planificateur de tâche

Nom du traitement	Fréquence	Heure	Date de première exécution

Proposition de corrigé

Travail à faire
1 – Commenter chaque ligne de l'ébauche de script Perl fournie en annexe 1.

```
# ! /usr/bin/perl
use strict ;

# récupération du numéro du jour de l'année
my $date = (localtime)[7] ;

# sauvegarde de la base de données zenta
system "pg_dump -i -h localhost -p 5432 -U tinypg -F c -v -f \"sauvegarde.dump.$date\" zenta ;
# fin de la sauvegarde

# transfert du fichier dump
system "ftp -i -n -v -s:ftp.txt"
# fin du transfert
```

Travail à faire
2 – Rajouter dans le script Perl la gestion de ce fichier de log (passage du nom par paramètre, ouverture écriture des messages de réussite et fermeture). Vous utiliserez l'annexe 2.

```
# ! /usr/bin/perl
use strict ;

# préparatifs on récupère le nom du fichier log passé en paramètre et le jour dans l'année
my $log = $ARGV[0];
my $date = (localtime)[7] ;

# ouverture du fichier log
open (F, "> $log") || die " Problème à l'ouverture du fichier : $!" ;
print F "La sauvegarde commence\n";

# sauvegarde
system "pg_dump -i -h localhost -p 5432 -U tinypg -F c -v -f \"sauvegarde.dump.$date\" zenta ;
print F "La sauvegarde s'est bien déroulée\n";

# transfert du fichier
system "ftp -i -n -v -s:ftp.txt ;
print F "Le transfert s'est bien déroulé\n" ;

close F ;
#fin
```

Travail à faire
3 – Rajouter dans le script Perl la gestion des codes de retour et des messages de réussite ou d'échec de chacune des étapes du traitement. Vous utiliserez l'annexe 3.

```
# ! /usr/bin/perl
use strict ;

# préparatifs on récupère le nom du fichier log passé en paramètre et le jour dans l'année
my $log = $ARGV[0];
my $date = (localtime)[7] ;

# ouverture du fichier log
open (F, "> $log") || die " Problème à l'ouverture du fichier : $!" ;
print F "La sauvegarde commence\n";
```

```
# sauvegarde
system "pg_dump -i -h localhost -p 5432 -U tinypg -F c -v -f \"sauvegarde.dump.$date\" zenta ;
if ($?==0)
        {
        print F "La sauvegarde s'est bien déroulée\n";
        # transfert du fichier
        system "ftp -i -n -v -s:ftp.txt ;
        # gestion des erreurs sur le transfert
        if ($?==0)
                {
                print F "Le transfert s'est bien déroulé\n" ;
                }
                else
                {
                print F "Il y a un problème sur le transfert" ;
                }
        }
        else
        {
        print F "Il y a un problème sur la sauvegarde" ;
        }
close F ;
#fin
```

Travail à faire
4 – Rajouter dans le script Perl la gestion des fichiers de log de chacune des étapes.

```
# ! /usr/bin/perl
use strict ;

# préparatifs on récupère le nom du fichier log passé en paramètre et le jour dans l'année
my $log = $ARGV[0];
my $date = (localtime)[7] ;

# ouverture du fichier log
open (F, "> $log") || die " Problème à l'ouverture du fichier : $!" ;
print F "La sauvegarde commence\n";

# sauvegarde
system "pg_dump -i -h localhost -p 5432 -U tinypg -F c -v -f \"sauvegarde.dump.$date\" zenta 2>
pg_dumplog.txt";
if ($?==0)
        {
        print F "La sauvegarde s'est bien déroulée\n";
        # destruction des fichiers intermédiaires
        system "del pg_dumplog.txt" ;
        # transfert du fichier
        system "ftp -i -n -v -s:ftp.txt > ftplog.txt" ;
        # gestion des erreurs sur le transfert
        if ($?==0)
                {
                print F "Le transfert s'est bien déroulé\n" ;
                # destruction des fichiers intermédiaires
                system "del sauvegarde.dump.$date ftplog.txt" ;
                }
                else
                {
                print F "Il y a un problème sur le transfert" ;
                # on conserve le fichier ftplog.txt pour analyse
                }
```

```
        }
        else
        {
        print F "Il y a un problème sur la sauvegarde" ;
        # on conserve le fichier pg_dumplog.txt pour analyse
        }
close F ;
#fin
```

Travail à faire
5 – Planifier ce traitement. Vous utiliserez l'annexe 4.

Nom du traitement	Fréquence	Heure	Date de première exécution
Sauvegarde.pl	Quotidienne	23 h 00	Aujourd'hui